

# CSC 413-03

## Software Development

### Term Project

#### 1. Tank Game

<https://github.com/csc413-03-fall2019/csc413-tankgame-YGLEEE1993>

#### 2. Super Rainbow Reef

<https://github.com/csc413-03-fall2019/csc413-secondgame-YGLEEE1993>

YUGYEONG(YG) LEE

918291951

Fall 2019

# Table of Contents

|           |   |           |
|-----------|---|-----------|
| <b>1</b>  | <b>Introduction.....</b>  | <b>4</b>  |
| 1.1       | Project Overview.....   | 4         |
| 1.2       | Introduction of the Tank Game .....   | 4         |
| 1.3       | Introduction of the Second Game.....  | 4         |
| <b>2</b>  | <b>Development environments .....</b>   | <b>5</b>  |
| 2.1       | Version of Java Used.....   | 5         |
|           | Java version "1.8.0_201" .....  | 5         |
| 2.2       | IDE Used .....  | 5         |
| <b>3</b>  | <b>How to Build/Import your Project.....</b>  | <b>5</b>  |
|           | Launch IntelliJ IDEA -> Import my project or use command line " java -jar [path to the jar file] "..... | 5         |
| <b>4</b>  | <b>How to Run your Project .....</b>  | <b>5</b>  |
| 4.1       | How to create Jar File .....  | 5         |
| 4.2       | Commands needed to run the build jar .....  | 5         |
| <b>5</b>  | <b>Rules and Controls of the game.....</b>  | <b>6</b>  |
| 5.1       | Tank Game Rules and Controls.....   | 6         |
| 5.2       | Super Rainbow Reed Rules and Controls .....   | 6         |
| <b>6</b>  | <b>Assumptions made when designing and implementing both games.....</b>                                 | <b>7</b>  |
| <b>7</b>  | <b>Tank Game Class Diagram .....</b>  | <b>8</b>  |
| <b>8</b>  | <b>Second Game Class Diagram .....</b>  | <b>9</b>  |
| <b>9</b>  | <b>Class Descriptions of all classes shared among both games .....</b>                                  | <b>10</b> |
| 9.1       | Launcher Class .....  | 10        |
| 9.2       | Display .....   | 10        |
| 9.3       | ImageLoader.....  | 12        |
| 9.4       | SpriteSheet .....   | 12        |
| 9.5       | Assets.....   | 13        |
| 9.6       | Game Class .....  | 14        |
| 9.7       | GameWorld Class .....   | 16        |
| <b>10</b> | <b>Class Descriptions of Tank Game.....</b>   | <b>17</b> |
| 10.1      | Bullet Class .....  | 17        |
| 10.2      | Controller Class.....   | 17        |
| 10.3      | Entity Class & EntityManager Class.....   | 17        |
| 10.4      | Item Class & ItemManager Class.....   | 18        |
| 10.5      | Handler Class.....  | 18        |

|           |  |           |
|-----------|--|-----------|
| 10.6      | <i>Tile Class</i> .....                        | 19        |
| <b>11</b> | <b>Class Descriptions of Second Game</b> ..... | <b>20</b> |
| 11.1      | <i>Menu Class</i> .....                        | 20        |
| 11.2      | <i>Block Class</i> .....                       | 21        |
| 11.3      | <i>KeyManager Class</i> .....                  | 21        |
| <b>12</b> | <b>Self-reflection</b> .....                   | <b>22</b> |
| <b>13</b> | <b>Project Conclusion</b> .....                | <b>22</b> |

# 1 Introduction

## 1.1 Project Overview

For the term project in CSC413, we make two 2D games written in Java, a presentation, and documentation file for the whole term project. I created a Tank Wars game and Rainbow Reef. The goal of this project is to practice good OOP which we focused on practicing good reusability for design and implantation of the both games.

## 1.2 Introduction of the Tank Game

First game required us to implement Tank Wars Game in Java. Sample code and resources were provided but we were free to use our own resources as well.

The requirements were:

1. Tank Game must have 2 Players
2. Tank Game must have tanks that move forwards and backwards
3. Tank Game Must have tanks that rotate so they can move in all directions
4. Tank Game must have a split screen
5. Tank Game must have a mini-map
6. Tank Game must have health bars for each tank
7. Tank Game must have lives count (how many lives left before game over) for each tank
8. Tank Game must have power up (these are items that can be picked up to modify your tank. What these power ups are, is up to you)
9. Tank game must have unbreakable walls
10. Tank game must have breakable walls
11. Tank Game must have tanks that can shoot bullets that collide with walls
12. Tank Game must have tanks that can shoot bullets that collide with other tanks.
13. Tank Game must come with a brief readme.txt file
14. 14. Tank Game must be built into a JAR and stored into the correct folder in the GitHub repo. This will be the folder named jar

## 1.3 Introduction of the Second Game

For the second game we were required to implement other game based on implementation of the Tank Game (such that the second game should take less time).

I choice Super Rainbow Reef for the second game. It is a one player brick breaker game.

The requirements were:

1. Super Rainbow Reef Game have several levels
2. Must have 3 lives
3. Game gets faster
4. ... more

## 2 Development environments

### 2.1 Version of Java Used

Java version "1.8.0\_201"

### 2.2 IDE Used

IntelliJ

## 3 How to Build/Import your Project

Launch IntelliJ IDEA -> Import my project or use command line “ **java -jar [path to the jar file]** ”

## 4 How to Run your Project

### 4.1 How to create Jar File

Go to File -> Project Structure -> Artifacts and add your project.

In your project -> out -> artifacts -> csc413\_secondgame\_YGLEE1993\_jar -> right click on jar file “*csc413\_secondgame\_YGLEE1993\_jar*” -> copy path.

### 4.2 Commands needed to run the build jar

In terminal type “ **java -jar [path to the jar file]** ”

## 5 Rules and Controls of the game

### 5.1 Tank Game Rules and Controls



Keys to move the Tank:

A: Move Left

D: Move Right

W: Move Top

S: Move Bottom

LEFT ARROW KEY: Attack

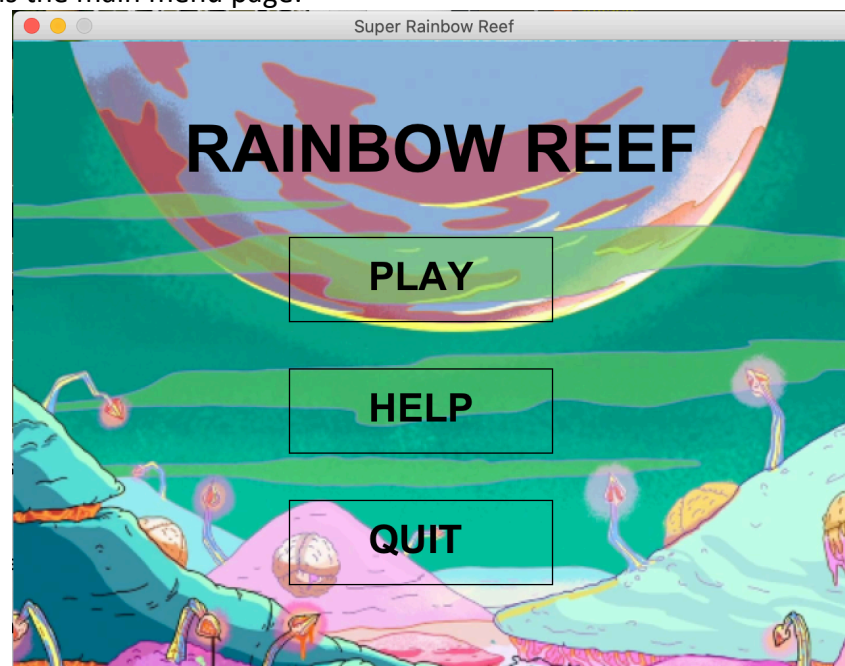
Rules for the Tank game is to destroy player 2 and break breakable walls. I have not completed to have 2 players and working bullet collision detection.

### 5.2 Super Rainbow Reef Rules and Controls

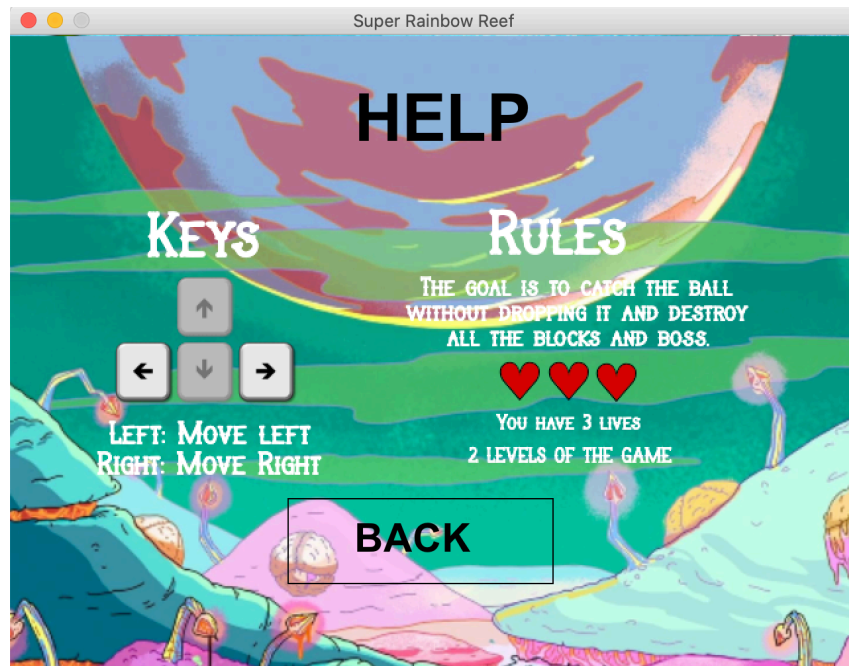
Player can press their mouse on the “PLAY” button to start play the game.

To see simple explanation of the game rules and controls, you can click on the “HELP” Button and to exit the game press the “QUIT” button.

Here is the main menu page:



When you click “HELP” button the below page is going to show and you can press “BACK” button to go back to the main menu.

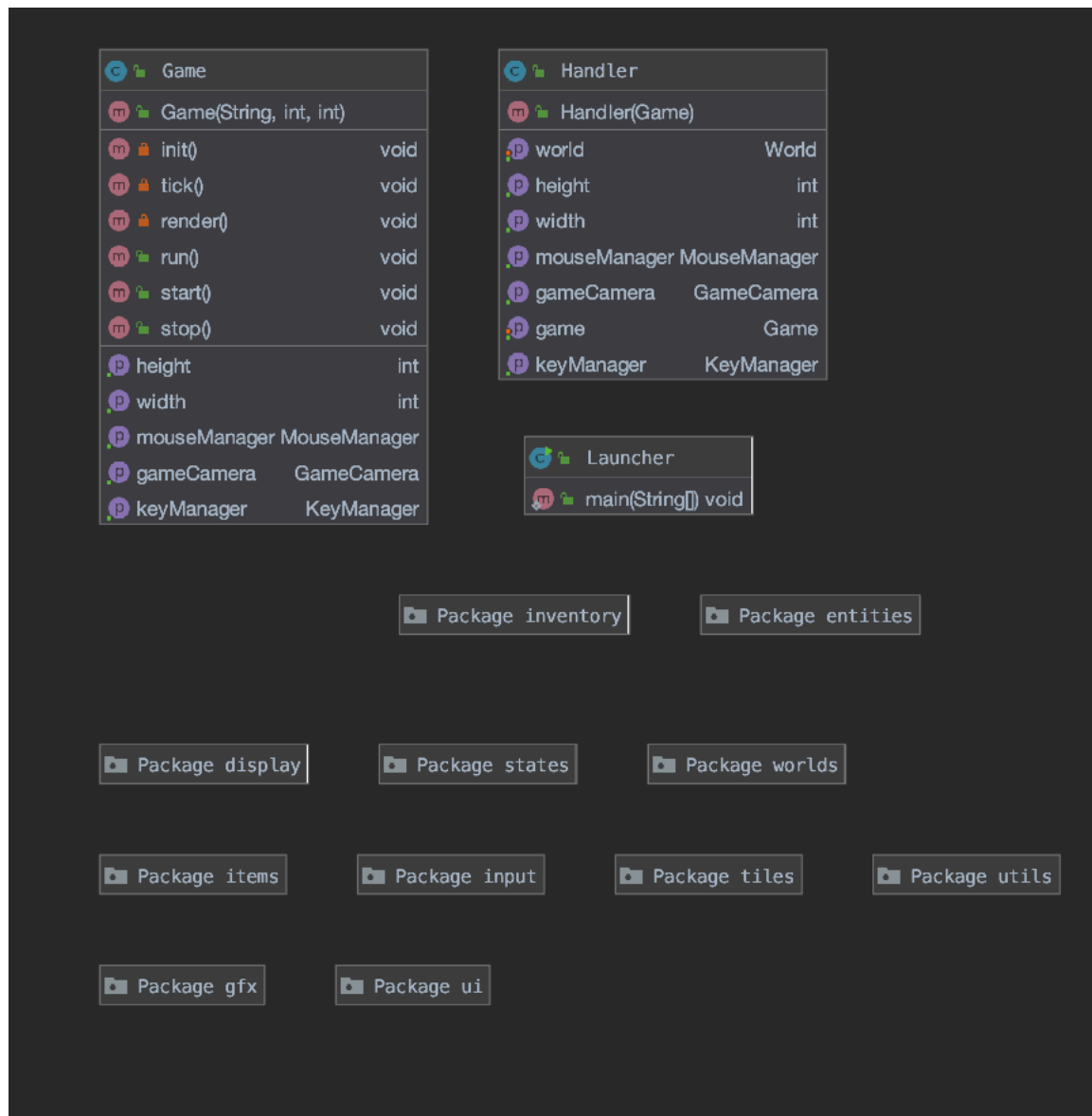


The player has 3 lives which can be affected by dropping the ball. There are 2 Levels of this game: Level 1 is easy version and Level 2 is harder map setting.

## 6 Assumptions made when designing and implementing both games

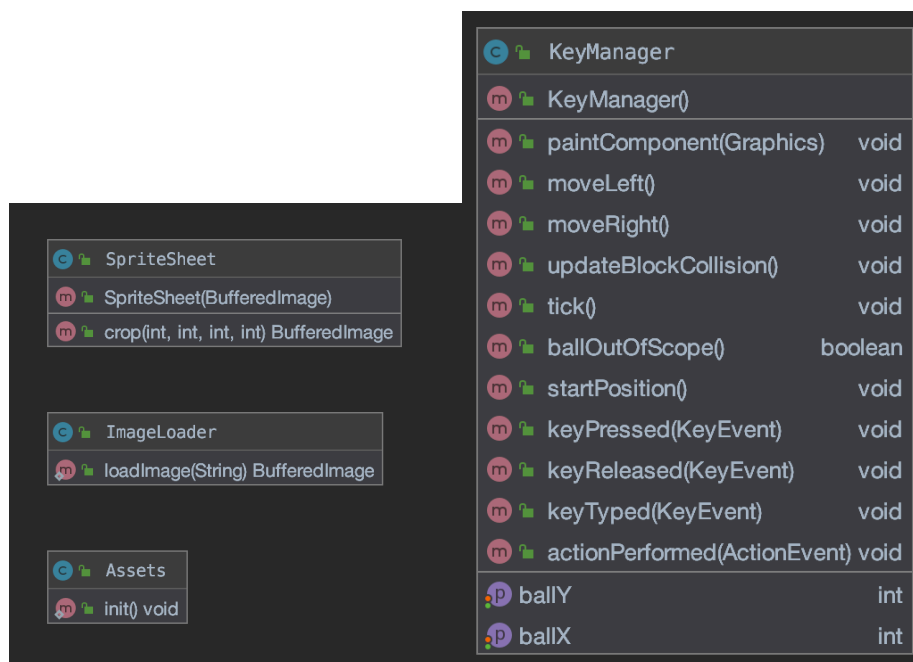
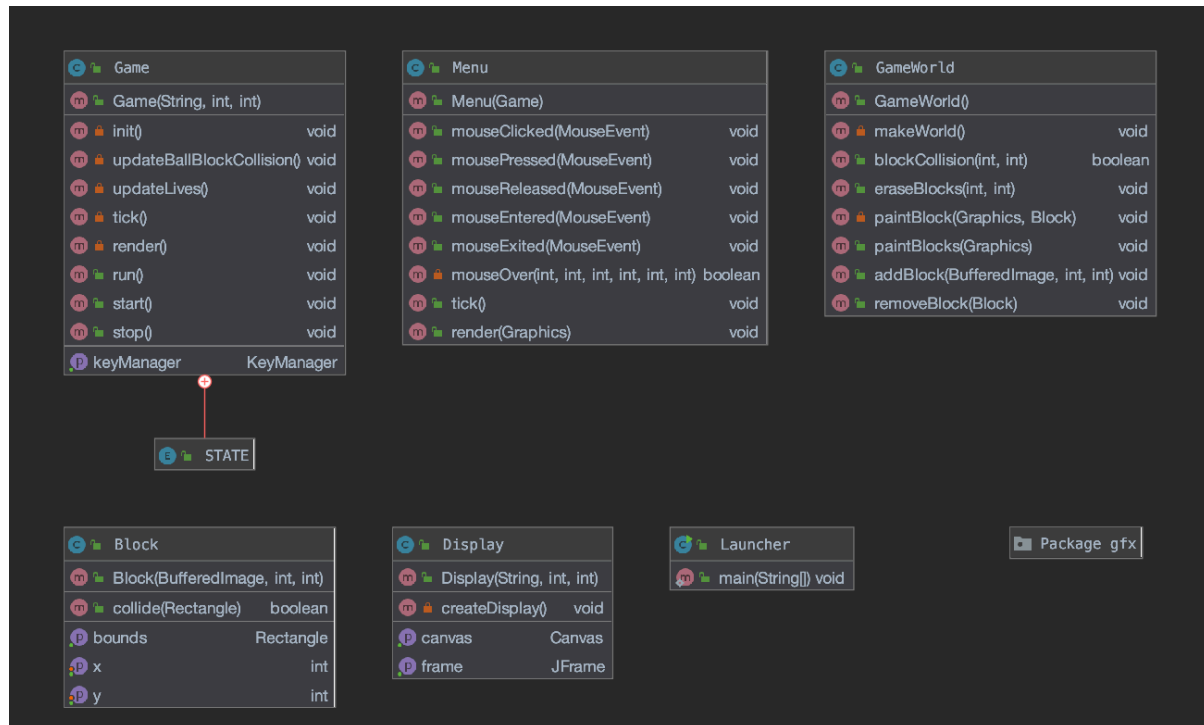
Even before trying to design the game, I had no idea where to start. I spent lots of time watching tutorials on java games.

## 7 Tank Game Class Diagram



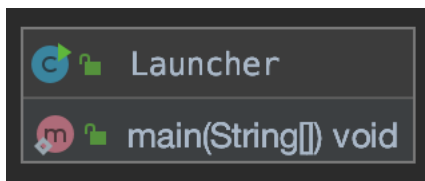



## 8 Second Game Class Diagram



## 9 Class Descriptions of all classes shared among both games

### 9.1 Launcher Class



|   |                     |
|---|---------------------|
|  | Launcher            |
|  | main(String[]) void |

Launcher Class is basically what it starts the Tank Game. It is the *main(String[]) void* class that implemented Game Object called game and it calls the *Start( )* method.








Tank Game:

```
public static void main(String[] args){  
    Game game = new Game( title: "Tank Game", width: 640, height: 480);  
    game.start();  
}
```

Super Rainbow Reef:

```
public static void main(String[] args){  
    Game game = new Game( title: "Super Rainbow Reef", width: 640, height: 480);  
    game.start();  
}
```

### 9.2 Display

|   |                           |        |
|---|---------------------------|--------|
|  | Display                   |        |
|  | title                     | String |
|  | width                     | int    |
|  | height                    | int    |
|  | Display(String, int, int) |        |
|  | createDisplay()           | void   |
|  | canvas                    | Canvas |
|  | frame                     | JFrame |

Display Class makes sure the project is displayable. It uses JFrame which is the window of the game and Canvas which is the paint of the game.

The constructor calls the *createDisplay( )*

```
//Constructor
public Display(String title, int width, int height) {
    this.title = title;
    this.width = width;
    this.height = height;
    createDisplay();
}
```

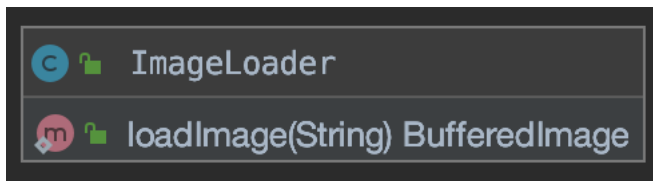
The createDisplay() sets the title, size of the window and properties and adds to the JFrame and Canvas.

```
private void createDisplay(){
    frame = new JFrame(title);
    frame.setSize(width, height);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setResizable(false); // not resizable
    frame.setLocationRelativeTo(null); // display center of the screen
    frame.setVisible(true); // default = false;

    canvas = new Canvas();
    canvas.setPreferredSize(new Dimension(width, height));
    canvas.setMaximumSize(new Dimension(width, height));
    canvas.setMinimumSize(new Dimension(width, height));
    canvas.setFocusable(false); // let JFrame focus only

    frame.add(canvas);
    frame.pack(); // adjust to show the canvas better
}
```

### 9.3 ImageLoader

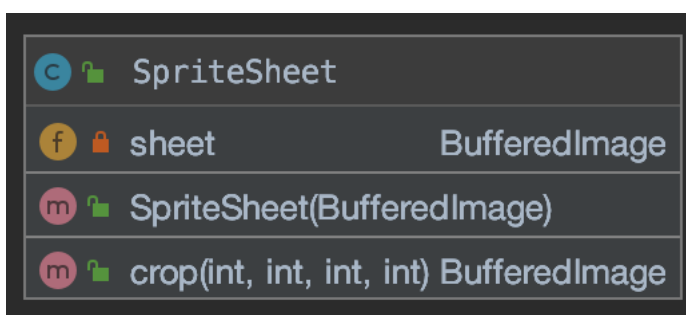


ImageLoader Class basically loads Image file.

```
public class ImageLoader {  
    public static BufferedImage loadImage(String path){  
        try {  
            return ImageIO.read(ImageLoader.class.getResource(path));  
        } catch (IOException e) {  
            e.printStackTrace();  
            System.exit( status: 1);  
        }  
        return null;  
    }  
}
```

ImageLoader Class has a loadImage(String) function takes the path to the Image file and returns buffered image.

### 9.4 SpriteSheet



SpriteSheet takes sprite sheet and crops them as I needed.

```

public class SpriteSheet {
    private BufferedImage sheet;

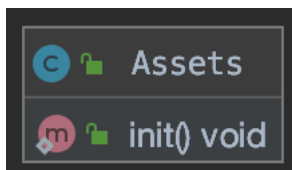
    public SpriteSheet(BufferedImage sheet){
        this.sheet = sheet;
    }

    public BufferedImage crop(int x, int y, int width, int height){
        return sheet.getSubimage(x, y, width, height);
    }
}

```

Crop ( ) takes 4 parameter that represents the dimension of the sprite sheet and returns the cropped BufferedImage by using getSubimage() method.

## 9.5 Assets



To avoid rendering all the assets (basically any types of image or texture) for the game every time Game class calls the render method, Assets class is basically has all the image files for the game assets.

```

public static void init(){
    SpriteSheet sheet = new
    SpriteSheet(ImageLoader.loadImage("/sheet.png"));

    bullet = sheet.crop(width * 2, height * 2, width, height);
}

```

Init ( ) takes SpriteSheet witch calls the LoadImage ( ) and we set all the Image we need it to crop out.

## 9.6 Game Class

Game Class is uses Thread which is basically mini program that allow us to run the class separately than the rest of the program.

Both of the games use same feature of the Game class:

**Init( ):** It basically initializes everything we need in the Game class.  
It initializes the Display, Assets, Handler, GameCamera,add  
KeyListener, add mousetlistener.

**tick( ):** Updates everything that need in the game

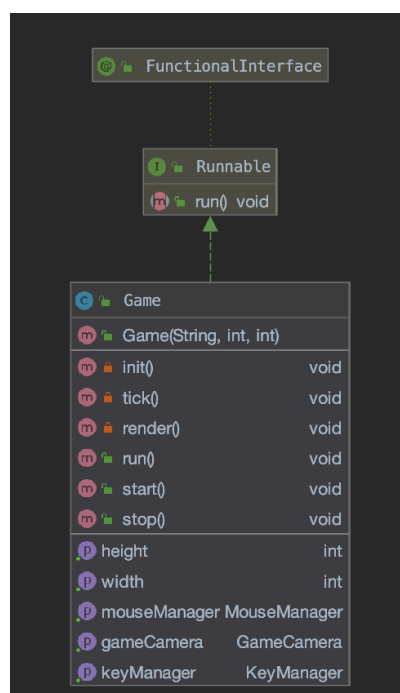
**render():** Draws everything to the screen with JFrame and Canvas.  
It uses the BufferedImage which is a way to draw something to  
the screen using buffer (A “hidden” computer screen).

**start( ):** It initializes the thread using synchronized and makes sure the game  
wasn't running when it's called and thread.start( ) is called which  
eventually calls the run ( ).

**stop( ):** It stops the game and makes sure that the thread wasn't stopped  
already and calls the thread.join( ).

**run( ):** It calls the init( ) and it has variables that keeps track of ticks per sec,  
timePerTick, amount of time we had until we have to call tick and  
render method again, current time of our computer, returns current  
time in this computer. Based on that it calls the tick( ) and render( ).

Tank Game:



### Super Rainbow Reef:



Additional to the Tank Game's Game Class, the Super Rainbow Reef game has:

**updateLives( ):** The second game has THREE lives and this function keep track of how many times the player died and based on that it draws and removes the lives count asset.

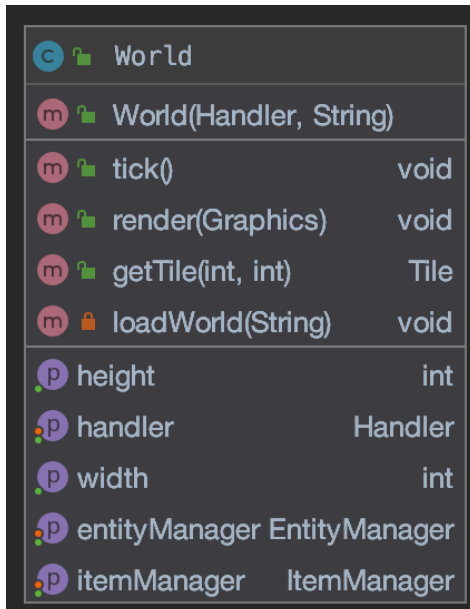
**updateBallBlockCollision( ):** It checks the blockCollision( ) which keeps track of if the ball hit the block and each time ball hits the block the score gets 1 point. It keeps track of the score and display it on the scree.

**public enum STATE:** enum for the game states.

## 9.7 GameWorld Class

The Idea of the GameWorld Class is to loadWorld for the game and it adds all the entities and textures that needed to present in the game screen.

### Tank Game:

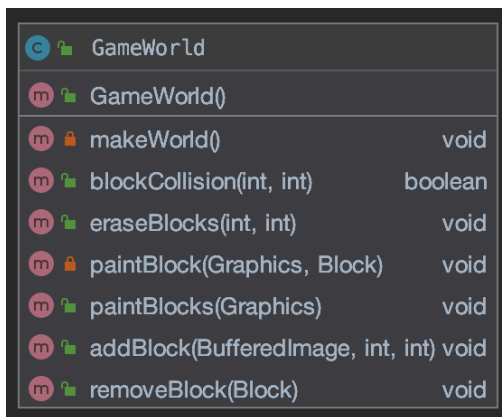


A screenshot of the 'World' class in an IDE. The class has a constructor 'World(Handler, String)' and methods 'tick()', 'render(Graphics)', 'getTile(int, int)', and 'loadWorld(String)'. It also has private fields 'height' (int), 'handler' (Handler), 'width' (int), 'entityManager' (EntityManager), and 'itemManager' (ItemManager).

| Icon        | Method/Field           | Return Type   |
|-------------|------------------------|---------------|
| Constructor | World(Handler, String) |               |
| Method      | tick()                 | void          |
| Method      | render(Graphics)       | void          |
| Method      | getTile(int, int)      | Tile          |
| Method      | loadWorld(String)      | void          |
| Field       | height                 | int           |
| Field       | handler                | Handler       |
| Field       | width                  | int           |
| Field       | entityManager          | EntityManager |
| Field       | itemManager            | ItemManager   |

Tick ( ): ticks the EntityManager  
ItemManager  
Controller  
Render ( ): renders ItemManger  
EntityManager  
getTile ( ): gets all the background texture  
I used for Tank Game which is  
Tiles.  
loadWolrd ( ): double for loop assigns the  
titles[] [] 2D array and  
creates the world based  
on the would.txt file.

### Super Rainbow Reef:



A screenshot of the 'GameWorld' class in an IDE. The class has a constructor 'GameWorld()' and methods 'makeWorld()', 'blockCollision(int, int)', 'eraseBlocks(int, int)', 'paintBlock(Graphics, Block)', 'paintBlocks(Graphics)', 'addBlock(BufferedImage, int, int)', and 'removeBlock(Block)'. It has no private fields.

| Icon        | Method/Field                      | Return Type |
|-------------|-----------------------------------|-------------|
| Constructor | GameWorld()                       |             |
| Method      | makeWorld()                       | void        |
| Method      | blockCollision(int, int)          | boolean     |
| Method      | eraseBlocks(int, int)             | void        |
| Method      | paintBlock(Graphics, Block)       | void        |
| Method      | paintBlocks(Graphics)             | void        |
| Method      | addBlock(BufferedImage, int, int) | void        |
| Method      | removeBlock(Block)                | void        |

makeWorld ( ): Using for loops I assign the  
blocks and boss for the game.  
blockCollision ( ): Using Rectangle  
getBounds ( ) and intersects ( ) to  
check the block collision.  
eraseBlock ( ): removes the blocks  
paintBlocks ( ): it calls the  
paintBlock ( ) and paints the block  
addBlock ( ) : add block  
removeBlock ( ): It checks if the all the  
blocks and boss are destroyed  
and if you finish level1 than it  
shows the Dialog that gives you  
an option to quit or play Level 2.

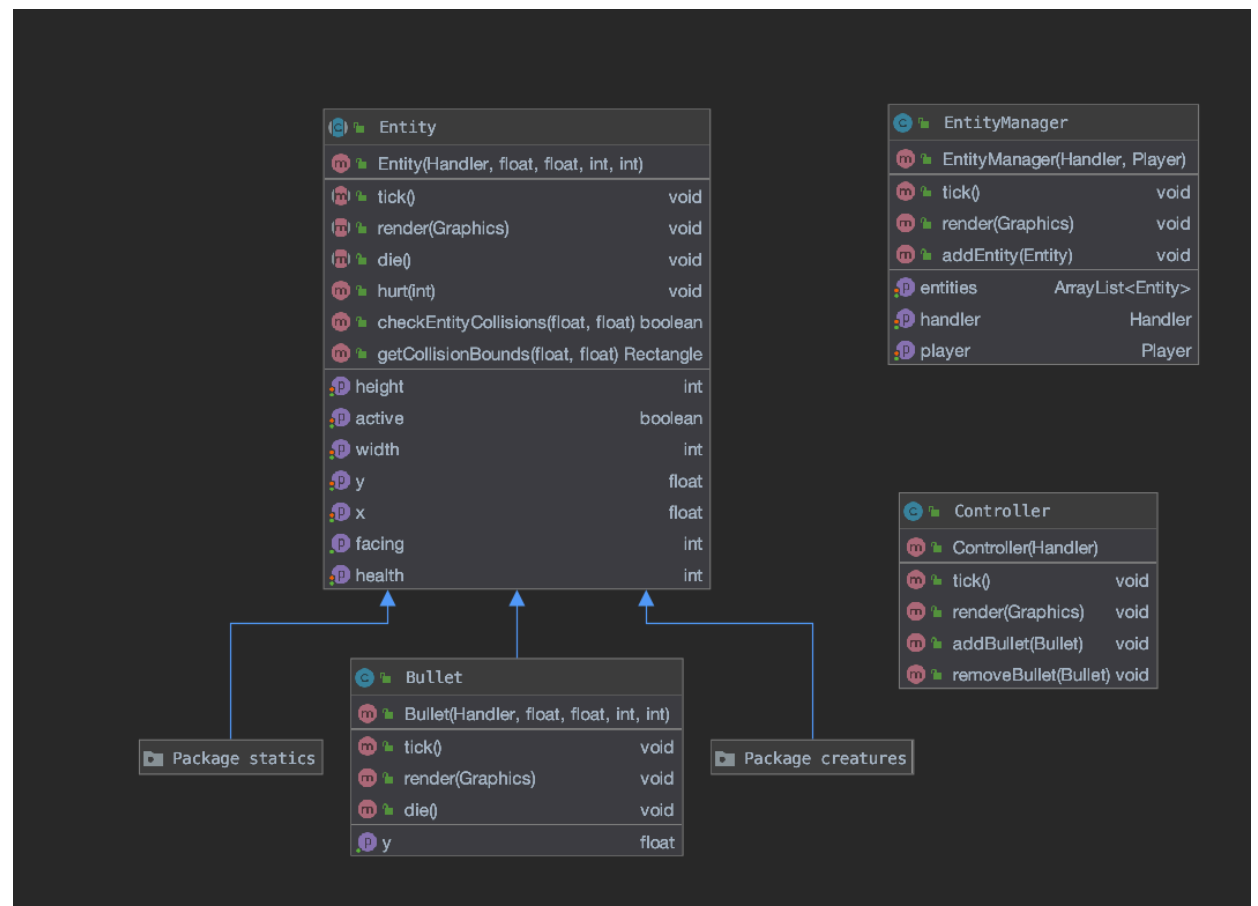


## 10 Class Descriptions of Tank Game

10.1 Bullet Class

10.2 Controller Class

10.3 Entity Class & EntityManager Class



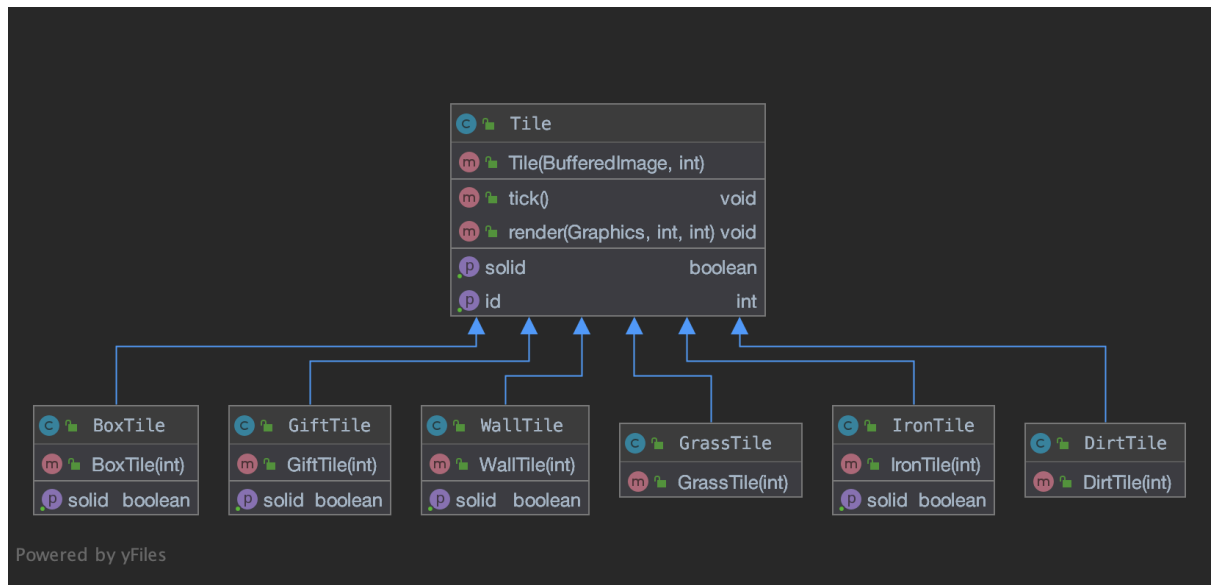
## 10.4 Item Class & ItemManager Class

| Item                                      | ItemManager                    |
|---|--------------------------------|
| <b>m</b> Item(BufferedImage, String, int) | <b>m</b> ItemManager(Handler)  |
| <b>m</b> tick() void                      | <b>m</b> tick() void           |
| <b>m</b> render(Graphics) void            | <b>m</b> render(Graphics) void |
| <b>m</b> render(Graphics, int, int) void  | <b>m</b> addItem(Item) void    |
| <b>m</b> createNew(int, int) Item         | <b>p</b> handler Handler       |
| <b>m</b> setPosition(int, int) void       |                                |
| <b>p</b> name String                      |                                |
| <b>p</b> pickedUp boolean                 |                                |
| <b>p</b> handler Handler                  |                                |
| <b>p</b> x int                            |                                |
| <b>p</b> count int                        |                                |
| <b>p</b> id int                           |                                |
| <b>p</b> y int                            |                                |
| <b>p</b> texture BufferedImage            |                                |

## 10.5 Handler Class

| Handler                            |
|------------------------------------|
| <b>m</b> Handler(Game)             |
| <b>p</b> world World               |
| <b>p</b> height int                |
| <b>p</b> width int                 |
| <b>p</b> mouseManager MouseManager |
| <b>p</b> gameCamera GameCamera     |
| <b>p</b> game Game                 |
| <b>p</b> keyManager KeyManager     |

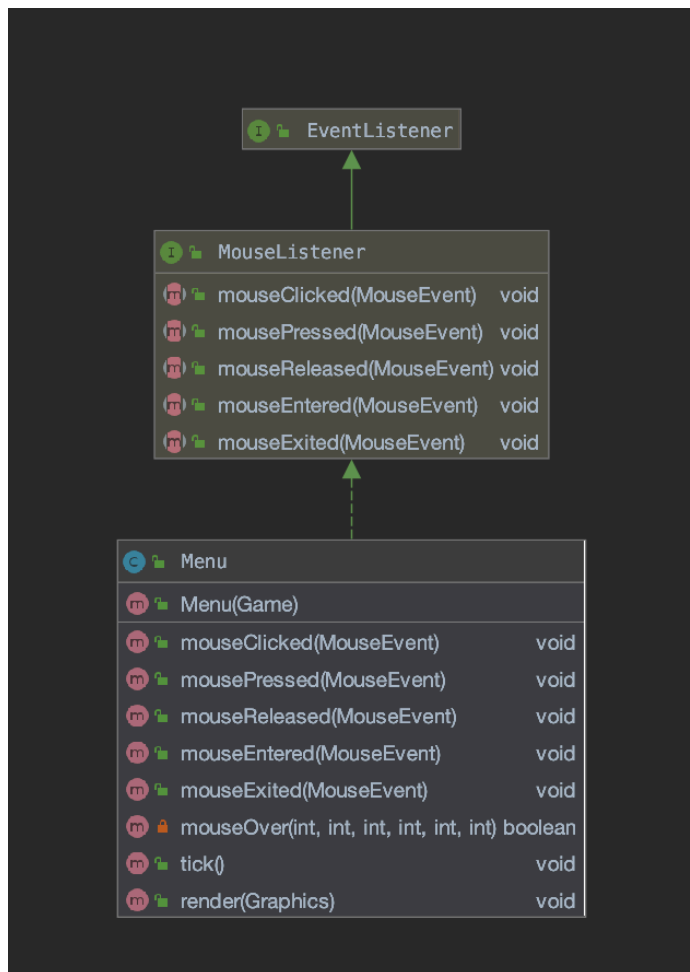
## 10.6 Tile Class



Tile Class is literally for all the tiles for the game. Parent Class **Tile** has all the basic features of what a tile should have. And so, when creating, adding, or removing tiles, it's much more efficient to create by extending the "Tile" parent class. Because all the features that tiles should have are common.

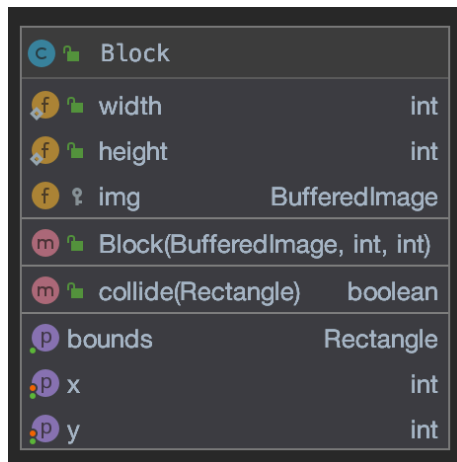
## 11 Class Descriptions of Second Game

### 11.1 Menu Class



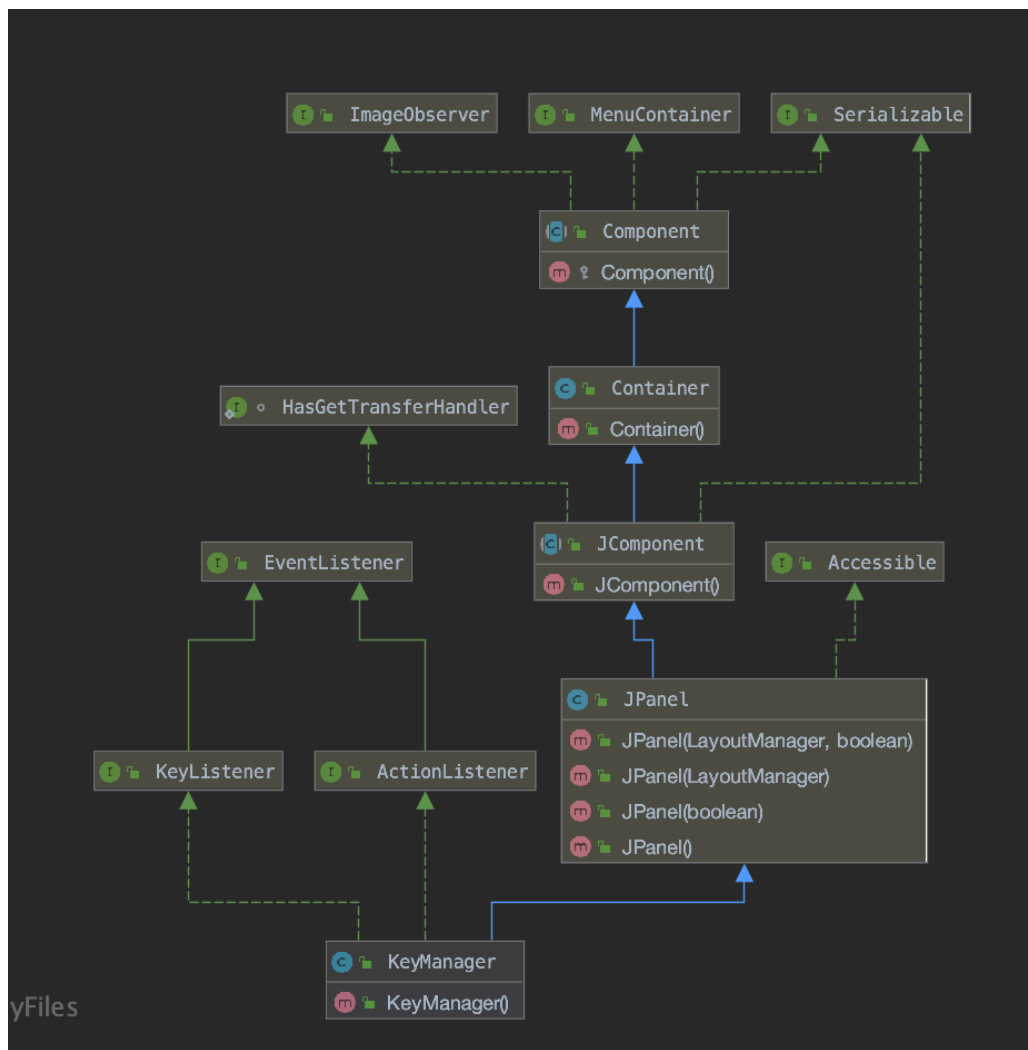
Menu Class is the first thing you see when you start the game. It has PLAY, HELP and QUIT buttons. If you click PLAY button it will start the game, HELP button it shows you basic rules and keys and QUIT button literally terminates the game.

## 11.2 Block Class



Block Class is literally for the block. It has it's dimension and functions to check the bound and collide.

## 11.3 KeyManager Class



KeyManager actually does more than what name represent. It extends JPanel and Implements KeyListener and ActionListener. It paints the ball and player assets and has functions that makes the ball moves and the player move to Left and Right. It has ballOutOfScope( ) that checks if the player has failed to catching the ball. Most importantly it makes sure what the game should do when the specific keys are pressed.

## 12 Self-reflection

This project was challenging but very rewarding. I have never made a game with java and when I started the tank game I struggled figuring out where to start and planning and designing the program. I could not complete the project requirement for Tank Game but am going to work on it since making two 2D games provided me much motivation to try more. Also, when making the Second game it was much easier planning and designing the game. Since I have already created the basic frame of what game should have, it took a shorter time to complete the Super Rainbow Reef game.

## 13 Project Conclusion

For this term projects, I had no experience developing a java game, but I learned so much through trying to build a game from almost scratch. Even though I have not completed Tank Game, it was such a good learning experience. I was able to develop the second game with better understanding of reusability.